AMENDMENTS TO THE SPECIFICATION

Please amend the paragraph beginning on page 7, line 17, as follows:

As routine names, variable names, and the like may be easily modified in a superficial manner, yet functionally remain the same, the present invention looks past the arbitrarily assigned labels in an executable script 208, and instead looks at its functional contents in a normalized form. A normalization module [[204]] 202 normalizes the executable script 208. A more detailed description of normalizing an executable script, i.e., the process by which an executable script is normalized, is described below. After normalizing an executable script 208, the normalization module [[204]] 202 outputs a script signature 210, the script signature corresponding to the normalized, functional contents of the executable script. In fact, as will be described in greater detail below, the normalization module [[204]] 202 performs two normalizations resulting in a first script signature 210 and a second script signature 212.

Please amend the paragraph beginning on page 9, line 11, as follows:

Those skilled in the art will recognize that most scripts include a "main" code segment. The main code segment may be located at the start of the script, such as is the case with Visual Basic® script files, or in some other location, often with a label of "main." The main code segment is that body of code from an executable script which is first executed. It should be noted that while this main code segment is often not considered to be a "routine," for purposes of the present invention, the main code segment may be treated as a "routine." Accordingly, at block 404, the first routine in the executable script 208 is selected.

Please amend the paragraph beginning on page 12, line 16, as follows:

If, at decision block 528, the routine token is not a subroutine token, at block 538, the routine token is written to the routine token set. Alternatively (not shown), other processing on the routine token may be made. For example, with respect to script languages that are case insensitive, such as Microsoft Corporation's Visual Basic ~~Script~~ script language, a routine token may be converted to a predetermined format, such as converting the characters of the routine token to lower case, in order to facilitate subsequent comparisons. Additional forms of processing may also be desirable. After writing the routine token to the routine token set, the process 500 returns again to decision block 512 (FIGURE 5A), where a determination is made as to whether there are any additional routine tokens to be processed in the current selected routine. This process 500 continues until, at decision block 512, the determination is made that there are no more routine tokens to be processed in the currently selected routine. Thereafter, at block 516, the routine token set is returned, and the routine terminates.

Please amend the paragraph beginning on page 13, line 24, as follows:

According to one embodiment of the present invention, a system based subroutine, i.e., those subroutines supplied by the particular script language, such as Visual Basic script <u>language</u> or JavaScript<u>® script language</u>, while they may technically be subroutine calls, are not normalized. Normalization is not needed in this case because a call to a system supplied routine cannot be superficially renamed. As such, a call to a system supplied routine in the executable script 208 may be properly compared to a call to the same routine in a known malware script signature. As illustrated in FIGURE 6, routine tokens 610 and 612 represent calls to system supplied subroutines. As such, they are not normalized but are simply written into the normalized output 700 as routine tokens 704 and 706. Similarly, member names of system defined data structures are not normalized, but are simply transferred over to the normalized output 700.